Fourier Neural Operator: Learning Solution Operators in Spectral Space

Krishna Kumar

University of Texas at Austin

krishnak@utexas.edu

Overview

Outline

Learning Objectives

- Understand the connection between CNNs and kernel operators
- Master Fourier Transform fundamentals for neural operators
- Learn the FNO architecture: spectral convolution layers
- Implement FNO for 1D Burgers equation
- Apply FNO to 2D Darcy Flow problem
- Explore mesh independence and super-resolution capabilities

▸ Open Notebook: FNO

The Central Challenge

We've seen how DeepONet learns operators by decomposing them into branch-trunk architectures.

But there's a deeper question

What if **physics itself suggests the right representation**?

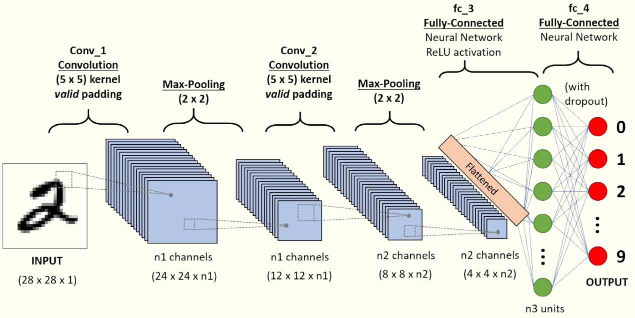
For 50+ years, **spectral methods** based on Fourier transforms have dominated computational physics because:

- Many PDEs simplify in Fourier space (convolution $\rightarrow$ multiplication)
- Derivatives become algebraic: $\mathcal{F}\left(\frac{\partial u}{\partial x}\right) = ik\hat{u}(k)$
- Global information propagates naturally

The FNO insight: Learn operators *in Fourier space* rather than physical space.

CNNs apply **local** kernels to extract features:

$$(f * g)(x) = \int_{\text{local}} f(x')g(x - x')dx'$$



CNN architecture with convolutional and pooling layers.

Key properties:

Kernel Operators: The General Form

A **kernel operator** maps functions to functions:

$$\mathcal{K}(v)(x) = \int_{\Omega} \kappa(x, x')v(x')dx'$$

where $\kappa(x, x')$ is a **learned kernel**.

Types of kernels:

1. **Standard convolution:** $\kappa(x, x') = k(x - x')$ (local, translation-invariant)
2. **Graph operators:** κ defined on graph edges
3. **Fourier operators:** κ learned in spectral space (global, efficient)

Why Fourier?

Convolution in physical space = multiplication in Fourier space!

Outline

The Fourier Transform

Mathematical Definition:

The Fourier Transform decomposes a function into sinusoidal components:

$$\hat{u}(k) = \mathcal{F}(u)(k) = \int_{-\infty}^{\infty} u(x)e^{-ikx}dx$$

Inverse Fourier Transform:

$$u(x) = \mathcal{F}^{-1}(\hat{u})(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{u}(k)e^{ikx}dk$$

Key insight: Any function can be written as:

$$u(x) = \sum_k \hat{u}_k e^{ikx}$$

where $\hat{u}_k$ are **Fourier coefficients** (complex weights) and e^{ikx} are basis functions.

Essential Properties of Fourier Transform

1. Derivatives become multiplication

$$\mathcal{F}\left(\frac{\partial u}{\partial x}\right) = ik\hat{u}(k)$$

2. Convolution becomes multiplication (Convolution Theorem)

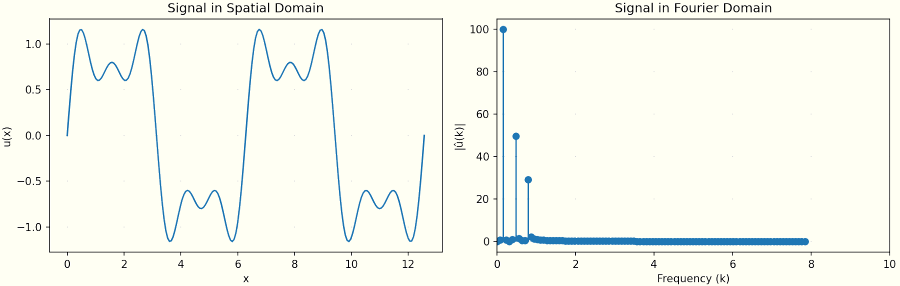
$$\mathcal{F}(u * v) = \mathcal{F}(u) \cdot \mathcal{F}(v)$$

3. Parseval's theorem (Energy conservation)

$$\int |u(x)|^2 dx = \int |\hat{u}(k)|^2 dk$$

Computational advantage: FFT is $O(N \log N)$, far cheaper than dense convolution $O(N^2)$.

Fourier Decomposition: Visualization



Fourier decomposition: A function (left) is represented as a sum of weighted sinusoids (middle), with most energy concentrated in low frequencies (right).

Key observation: Most energy concentrated in low frequencies.
FNO exploits this: Learn weights only for low-frequency modes ($k_{\text{max}} \approx 12 - 16$), discard high frequencies.

Outline

The Core Idea

Instead of learning in physical space, **learn in Fourier space**:

$$v_{t+1}(x) = \sigma\left(Wv_t(x) + \mathcal{K}(v_t)(x)\right)$$

where the kernel operator $\mathcal{K}$ is parameterized in Fourier space:

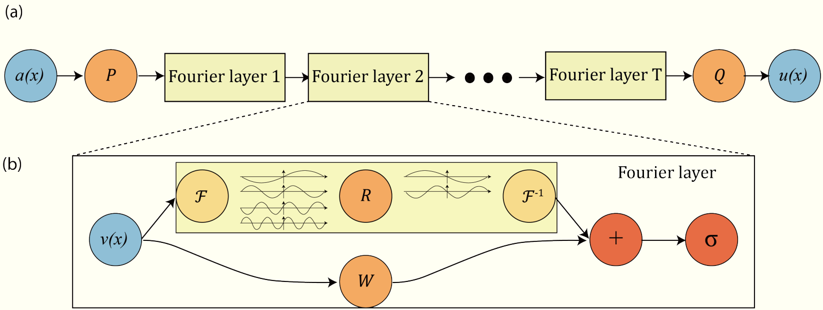
$$\mathcal{K}(v)(x) = \mathcal{F}^{-1}\left(R_\phi \cdot \mathcal{F}(v)\right)(x)$$

Here, R_ϕ are **learnable weights** in Fourier space.

The FNO workflow

$$\text{Input } u(x) \xrightarrow{\text{FFT}} \text{Fourier Space} \xrightarrow{\text{Learn } R_\phi} \text{Fourier Space} \xrightarrow{\text{IFFT}} \text{Output}$$

Complete FNO Architecture



Fourier Neural Operator architecture.

Architecture consists of:

- **Lifting:** $P(u) \rightarrow v_0$ (embed input to high-dimensional space)
- **Fourier Layers:** $v_{t+1} = \sigma(Wv_t + \mathcal{K}(v_t))$ (4 layers)
- **Projection:** $Q(v_4) \rightarrow$ output (map back to output space)

Spectral Convolution Layer

Each Fourier layer performs three operations:

1. **FFT:** $\hat{v} = \mathcal{F}(v)$
2. **Linear transform (truncated):**

$$\hat{v}_{\text{out}}[k] = R_\phi[k] \cdot \hat{v}[k] \quad \text{for } k \leq k_{\max}$$

3. **IFFT:** $v_{\text{out}} = \mathcal{F}^{-1}(\hat{v}_{\text{out}})$

Key design choice

Only keep low-frequency modes ($k_{\max} \approx 12 - 16$), discard high frequencies. **Why?** Most signal energy in low frequencies + acts as implicit regularization.

Spectral Convolution: Mathematical Details

For 1D problems:

$$\text{SpectralConv1d}(v) = \mathcal{F}^{-1}\left(R \cdot \mathcal{F}(v)[:k_{\max}]\right)$$

For 2D problems:

$$\text{SpectralConv2d}(v) = \mathcal{F}^{-1}\left(R_1 \cdot \mathcal{F}(v)[:k_1, :k_2] + R_2 \cdot \mathcal{F}(v)[-k_1 :, :k_2]\right)$$

Learnable parameters: $R \in \mathbb{C}^{c_{in} \times c_{out} \times k_{\max}}$

Skip connections: Standard 1x1 convolution in parallel

$$v_{\text{out}} = \sigma(\text{SpectralConv}(v) + Wv)$$

Outline

The 1D Viscous Burgers Equation

Problem Formulation:
The 1D viscous Burgers equation models shock wave propagation:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad x \in [0, 2\pi], t \in [0, T]$$

with periodic boundary conditions and initial condition $u(x, 0) = u_0(x)$.

Operator learning task: Learn the mapping

$$\mathcal{G} : u_0(x) \mapsto u(x, T)$$

from initial condition to solution at time $T = 1$.

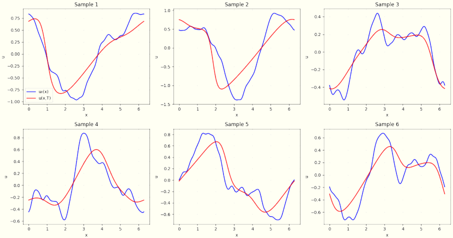
Dataset: 1024 training samples from varied initial conditions, solved using spectral methods.

The physics:

- **Nonlinear advection:** $u\frac{\partial u}{\partial x}$
- **Viscous diffusion:** $\nu\frac{\partial^2 u}{\partial x^2}$
- Competition creates shock waves
- Periodic boundary conditions

Why this problem:

- Prototype for nonlinear PDEs
- Captures shock formation
- Tests operator learning on complex dynamics



Sample initial conditions (blue) and evolved solutions (red) showing shock formation.

1D FNO Architecture and Training

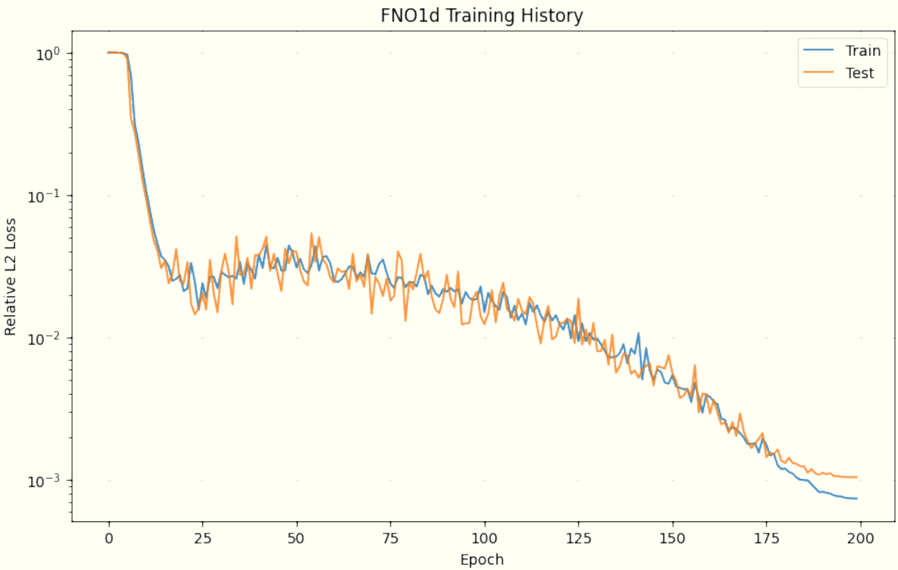
Model Configuration:

- **Input:** $(u_0(x), x)$ at $m = 2048$ points (subsampled to 512)
- **Fourier modes:** $k_{\max} = 16$
- **Hidden width:** $w = 64$
- **Layers:** 4 Fourier layers with skip connections
- **Activation:** GELU
- **Parameters:** $\sim$287K

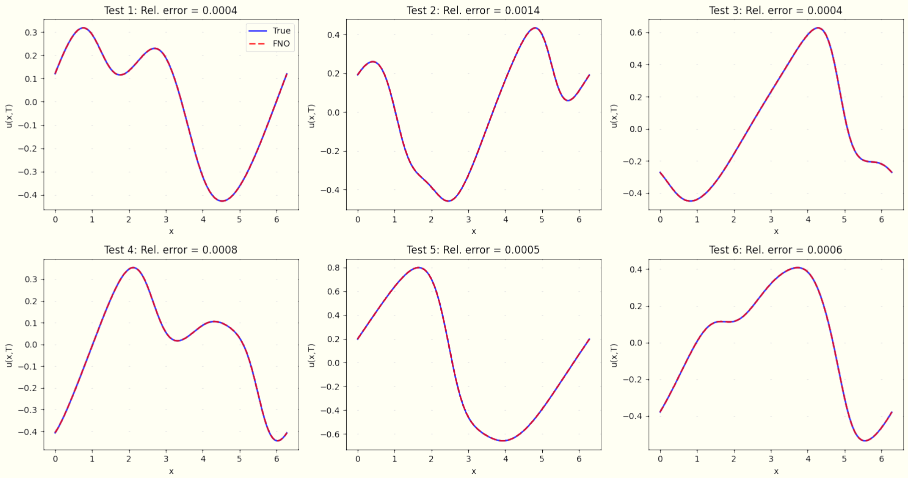
Training Configuration:

- **Loss:** Relative L^2 loss: $\frac{\|u_{pred} - u_{true}\|_2}{\|u_{true}\|_2}$
- **Optimizer:** Adam with OneCycleLR scheduler
- **Training samples:** 1000
- **Test samples:** 100
- **Epochs:** 200

Burgers Equation: Training Results



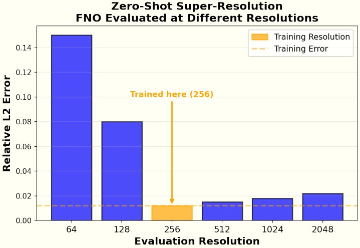
Burgers Equation: Prediction Quality

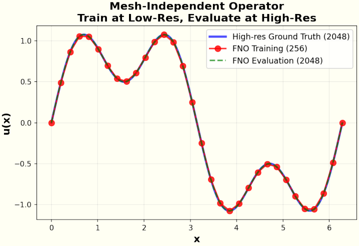


FNO predictions (red dashed) vs. true solutions (blue) for test cases.

Key results:

Zero-Shot Super-Resolution





Training at 512 resolution, evaluating at 2048 resolution (4x upsampling).

Mesh Independence

- **Train:** 512 grid points
- **Test:** 2048 grid points (4x refinement)
- **Result:** Maintains accuracy at higher resolution!

Why this works: FNO learns in Fourier space (continuous representation), not tied to specific discretization.

Outline

The 2D Darcy Flow Equation

Problem Formulation:
The 2D Darcy flow equation models steady-state flow in porous media:

$$-\nabla \cdot (a(x,y)\nabla u(x,y)) = f(x,y), \quad (x,y) \in [0,1]^2$$

with zero boundary conditions: $u|_{\partial\Omega} = 0$.

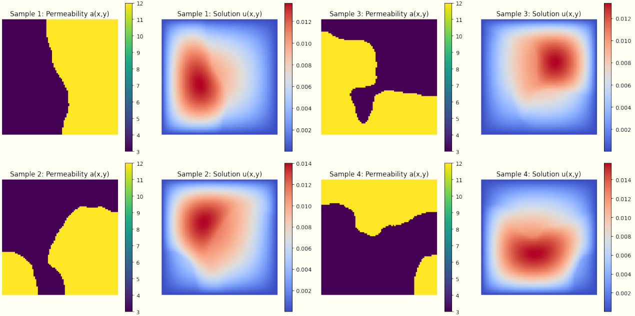
Operator learning task: Learn the mapping

$$\mathcal{G} : a(x,y) \mapsto u(x,y)$$

from permeability coefficient a to pressure/hydraulic head u.

Physical interpretation:

- $a(x,y)$: permeability field (how easily fluid flows)
- $u(x,y)$: pressure field
- $f(x,y)$: source term (set to 1)



Left: Permeability fields $a(x, y)$ (input). Right: Pressure fields $u(x, y)$ (output).

Dataset characteristics:

2D FNO Architecture

Model Configuration:

- **Input:** $(a(x,y), x, y)$ at 81×81 grid
- **Fourier modes:** $k_{\text{max}} = 12$ (both x and y directions)
- **Hidden width:** $w = 32$
- **Layers:** 4 Fourier layers
- **Parameters:** $\sim$130K

Spectral Convolution 2D:

$$\text{FFT2} \rightarrow \text{Multiply modes } [:12, :12] \text{ and } [-12:, :12] \rightarrow \text{IFFT2}$$

Why two weight matrices? Due to symmetry of real FFT (rfft2), we need weights for both lower and upper frequencies.

Darcy Flow: Training Configuration

Training Setup:

- **Loss:** Relative L^2 loss in 2D
- **Optimizer:** Adam with step decay (StepLR)
- **Learning rate:** 0.001 with decay every 50 epochs
- **Batch size:** 20
- **Epochs:** 200 (500 in original paper)

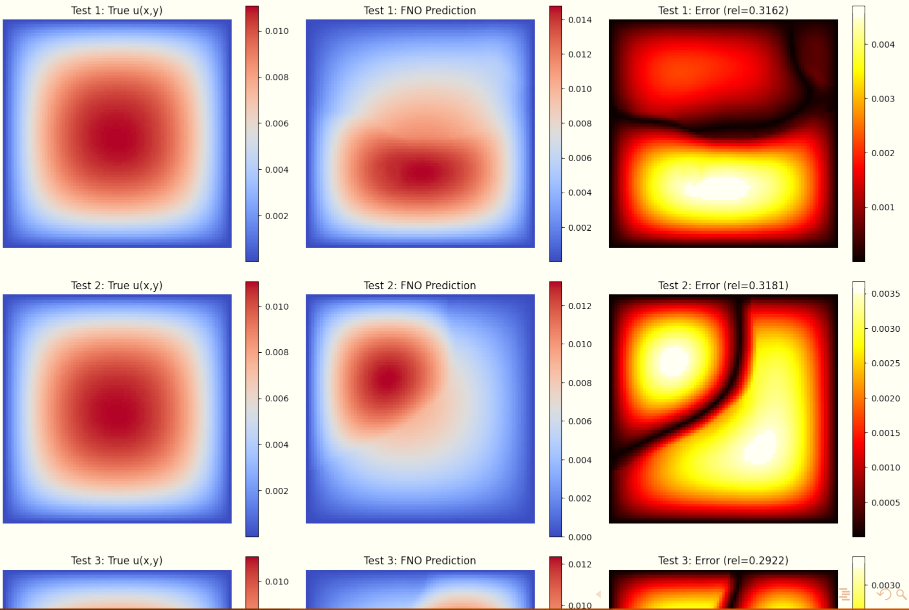
Data Normalization:

- Input permeability: Unit Gaussian normalization
- Output pressure: Unit Gaussian normalization
- Decode predictions before computing loss

Important

Normalize data for stable training, but always compute loss on physical quantities!

Darcy Flow: Results and Analysis



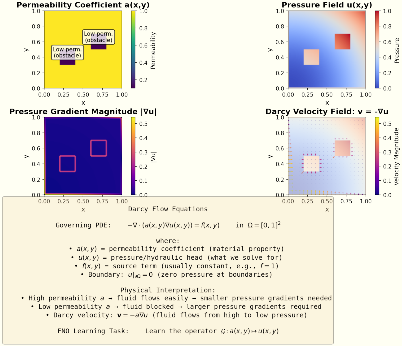
Understanding the "Two Bubbles"

Why circular patterns?

- Elliptic PDE creates smooth distributions
- Source term $f = 1$ uniformly
- Zero boundary conditions
- Low permeability $\rightarrow$ high pressure

Error patterns show:

- Higher errors near boundaries
- Errors in steep gradient regions
- Suggests need for more training



Physical interpretation: Permeability variations create pressure "bubbles".

Key takeaway: The patterns are physically correct - FNO learned the right physics!

Outline

What Makes FNO Special?

1. Mesh Independence

- Train on one resolution, evaluate on any resolution
- Works because we learn in Fourier space (continuous representation)
- Discretization is only for FFT computation

2. Computational Efficiency

- FFT: $O(N \log N)$ vs dense convolution $O(N^2)$
- Once trained: milliseconds per evaluation
- Traditional solver: seconds to minutes

3. Global Receptive Field

- Fourier modes capture global information
- No need to stack many layers for long-range dependencies
- Contrast with CNNs: local receptive fields

Physics-Informed Design

Why Fourier space is natural for physics:

- **50+ years of spectral methods:** Physicists have used Fourier for PDEs since the 1960s
- **Derivatives are algebraic:** $\partial_x \to ik$
- **Convolution theorem:** Simplifies nonlinear terms
- **Natural for periodic BCs:** Exact representation

Key insight

FNO doesn't invent a new representation - it uses the one physicists have known works well!

Low-frequency truncation benefits:

- **Efficiency:** Most energy in low frequencies
- **Regularization:** Implicit smoothing
- **Generalization:** Avoids overfitting to high-frequency noise

Limitations and Considerations

1. Periodic Boundary Conditions

- Standard FFT assumes periodicity
- Extensions needed for general geometries (Geo-FNO)
- Works perfectly for: Navier-Stokes on periodic domains
- Challenges for: Complex geometries, irregular domains

2. Data Requirements

- Need many solved PDE instances for training
- Expensive data generation phase
- Can be mitigated with physics-informed training

3. Black Box Nature

- No explicit PDE enforcement during training (standard FNO)
- May violate physical constraints
- Solution: Physics-Informed FNO

FNO vs DeepONet: Comparison

Aspect	DeepONet	FNO
Architecture	Branch-Trunk	Spectral Convolution
Space	Physical	Fourier
Queries	Arbitrary points	Grid points (FFT)
Best for	Irregular domains	Periodic domains
Parameters	More	Fewer
Speed	Fast	Faster (FFT)
Global info	Via trunk network	Natural in Fourier
Mesh free?	Yes	No (needs FFT grid)

Key difference:

- **DeepONet:** Learns basis decomposition in physical space
- **FNO:** Learns multiplication in Fourier space

Both are powerful! Choice depends on:

- Boundary conditions (periodic → FNO, irregular → DeepONet)
- Query pattern (grid → FNO, arbitrary points → DeepONet)

Outline

Extensions to FNO

Geo-FNO: Arbitrary Geometries

- Use coordinate transforms to handle non-periodic domains
- Applies Fourier layers in transformed space
- Enables FNO for complex geometries

Physics-Informed FNO

- Add PDE residual to loss function
- Ensures physical consistency
- Reduces data requirements

Factorized FNO

- Low-rank approximations for 3D problems
- Reduces memory and computation
- Enables high-resolution 3D operator learning

U-FNO: Multi-Scale Architecture

U-Net structure with Fourier layers:

- **Encoder:** Progressively downsample and increase channels
- **Fourier layers:** At each resolution level
- **Decoder:** Progressively upsample with skip connections

Advantages:

- Captures multi-scale physics
- Better for problems with localized features
- Skip connections preserve fine details

Applications:

- Turbulence (multi-scale eddies)
- Weather prediction (global + local patterns)
- Material design (micro + macro structures)

Practical Applications

Where FNO excels:

Fluid Dynamics:

- Navier-Stokes on periodic domains
- Turbulence modeling
- Weather/climate prediction
- Aerodynamics optimization

Scientific Computing:

- Quantum mechanics (Schrödinger)
- Wave propagation
- Heat diffusion
- Reaction-diffusion systems

Engineering Design:

- Structural optimization
- Material design
- Electromagnetics
- Acoustic modeling

Inverse Problems:

- Parameter identification
- Subsurface imaging
- Medical imaging
- Data assimilation

Outline

What We've Learned

1. Conceptual Foundation

- CNNs are local kernel operators
- FNO extends to global kernel operators in Fourier space
- Physics naturally lives in spectral domain

2. Mathematical Framework

- Fourier Transform decomposes functions into frequency components
- Convolution theorem: multiplication in Fourier space
- Operator parametrization: $\mathcal{K}(v) = \mathcal{F}^{-1}(R_\phi \cdot \mathcal{F}(v))$

3. Architecture

- Spectral convolution layers: FFT $\rightarrow$ Learn $\rightarrow$ IFFT
- Skip connections for expressivity
- Mode truncation for efficiency and regularization

Key Advantages of FNO

1. **Resolution independence:** Train on one grid, evaluate on any grid
2. **Computational efficiency:** $O(N \log N)$ via FFT
3. **Global receptive field:** Captures long-range dependencies naturally
4. **Physics-informed:** Leverages 50+ years of spectral methods
5. **Fast inference:** Once trained, millisecond evaluation
6. **Super-resolution:** Zero-shot upsampling capability

The paradigm shift

Traditional: Solve each PDE instance numerically
FNO: Learn the solution operator once, instant evaluation for any input

When to Use FNO

Ideal scenarios:

- **Periodic or translation-invariant problems**
- **Need for resolution independence**
- **Real-time PDE solving**
- **Large-scale parameter sweeps**
- **Multi-query problems:** Same operator, many evaluations

Consider alternatives when:

- **Complex, non-periodic geometries** (use Geo-FNO or DeepONet)
- **Sparse, irregular data** (DeepONet better)
- **Point-wise queries at arbitrary locations** (DeepONet)
- **Very limited training data** (consider PI-FNO)

Implementation Tips

1. Architecture Choices

- **Fourier modes:** Start with 12-16, adjust based on problem
- **Width:** 32-64 for most problems
- **Layers:** 4 is standard, more for complex operators
- **Activation:** GELU works well (smoother than ReLU)

2. Training Strategy

- **Loss:** Relative L^2 for scale-invariance
- **Optimizer:** Adam with learning rate scheduling
- **Normalization:** Unit Gaussian for inputs and outputs
- **Epochs:** 200-500 depending on problem complexity

3. Validation

- Test on different resolutions (super-resolution check)
- Verify physics consistency (if applicable)

Resources and Further Reading

Original Paper:

- Li et al., "Fourier Neural Operator for Parametric PDEs" (ICLR 2021)
- https://arxiv.org/abs/2010.08895

Extensions:

- Geo-FNO: https://arxiv.org/abs/2207.05209
- Physics-Informed Neural Operators: https://arxiv.org/abs/2111.03794
- Neural Operator Review: https://arxiv.org/abs/2108.08481

Code and Data:

- Official implementation: https://github.com/neuraloperator/neuraloperator
- Course notebooks: https://github.com/kks32-courses/sciml
- Dataset: https://huggingface.co/datasets/kks32/sciml-dataset

The Bigger Picture

From Functions to Operators:

- Neural Networks $\rightarrow$ Functions
- DeepONet $\rightarrow$ Operators (branch-trunk)
- FNO $\rightarrow$ Operators (spectral)
- Next: Multi-scale, multi-physics

Convergence of Fields:

- Machine Learning
- Numerical Analysis
- Applied Mathematics
- Scientific Computing

FNO represents a beautiful synthesis:

Using physics-inspired architectures (Fourier)

$+$

Machine learning (neural networks)

$=$

Fast, accurate operator learning for science and engineering

Questions?

Thank you!

Contact:

Krishna Kumar

krishnak@utexas.edu

University of Texas at Austin

Interactive Demo:

FNO Notebook